From Off-line to On-line Kidney Exchange Optimization

Danuta Sorina Chisca and Barry O'Sullivan
Insight Centre for Data Analytics
Dept. of CS, University College Cork, Ireland
{sorina.chisca|barry.osullivan}@insight-centre.org

Michele Lombardi and Michela Milano
DISI, University of Bologna, Bologna, Italy
{$michele.lombardi2$|$michela.milano$}@unibo.it

***Abstract*—Kidney exchange programs enable willing, but incompatible, donor-patient pairs to swap donors, thus allowing persons suffering from organ failure to access transplantation. Choosing which pairs to match requires to solve a stochastic on-line optimization problem where patients and donors arrive over time. Despite this, most of the related scientific literature has focused on deterministic off-line models. In this paper, we present a simple approach to employ a model for the off-line Kidney Exchange Problem (KEP) as the basis of an on-line anticipatory algorithm. Our approach grounds on existing techniques for the on-line KEP, but it generalise them and provides a more accurate estimate of the expected impact of current decisions. In an experimentation based on a state-of-the-art donor pool generation method, the approach provides improvements in terms of quality and is able to deal with realistic instance size in reasonable time.**

***Index Terms*—Optimisation, Anticipatory algorithm, Online stochastic kidney exchange.**

I. Introduction

Transplantation is an effective solution for people suffering of kidney failure [1], but finding a viable organ can be very difficult because of the scarcity of donors. Waiting for the organ of a deceased person can take more than a couple of years, and buying/selling of organs is illegal in most countries. However, people have two kidneys and a person can live fine with just one. This has encouraged voluntary donors (e.g. family members), which however are often incompatible with the patient due to (e.g.) blood or tissue type. A solution consists in having the incompatible pairs join a system that allows to swap donors, i.e. a kidney exchange program.

Centralised kidney exchange programs exist in many countries, including the US, the Netherlands and the UK [2], and require to solve at regular intervals an optimization problem (the Kidney Exchange Problem – KEP) to to find a matching for the enrolled pairs that saves the largest possible number of lives. This variant of the KEP has been addressed via a number of effective approaches. In reality however, new pairs arrive (and unfortunately drop-off) over time, making the problem *inherently on-line and stochastic*. In this setting, it is known that exploiting information about future outcomes, by means of an *anticipatory on-line algorithm*, can lead to substantially better results. The availability of extensive medical data makes the estimation of future outcomes practically viable.

On-line decision making can be framed as a form of multi-stage stochastic optimization. This class of problems is extremely hard to solve exactly, but high-quality solutions can be found via scalable sub-optimal methods (see [3] and references therein). These approaches typically estimate future developments by sampling *scenarios* and optimize the (estimated) expected value of current decisions. A few of the main anticipatory algorithms from [3] have been adapted and extended to the KEP in [4], but the task was not straightforward and forced the authors to introduce heuristic approximations.

Here, we present a method to build an on-line anticipatory algorithm from an off-line KEP model, with the objective to maximize the number of transplants. Compared to the algorithms from [4], our method is more general and *it correctly estimates expected values within the limit of sampling errors*. We call this method "Collective Scenario-Based Algorithm", for short CSBA. We compare this method against a baseline that reflects the current practice (i.e. solving the off-line model with the current pairs), against an oracle operating under perfect information, and against the best performing algorithms from [4]. On instances obtained via a realistic donor pool generation method, our approach improves over the state-of-the-art in terms of number of transplants.

The paper is organized as follows: Section II presents the related literature and the basics for understanding our algorithm, discussed in Section III. Experimental results are in Section IV and concluding remarks in Section V.

II. Problem Description and Related Work

Formally, the KEP is a non-bipartite matching problem on a directed graph, where nodes correspond to patient/donor pairs or simply to willing ("altruistic") donors. An arc from node i to j means that donor i is compatible with patient j, and a cycle (or chain starting from an altruistic donor) corresponds a viable set of exchanges. The goal is to maximize a utility function, in our case the total number of transplantations.

Figure 1 shows an example of such a graph, with four patient/donor pairs and one altruistic donor (i.e. $\langle -, d_4 \rangle$). Edges with two arrows corresponds to two arcs (one per direction). A two-way exchange is given by $d_0 \rightarrow p_2, d_2 \rightarrow p_0$, a three

This work has been supported by a research grant by Science Foundation Ireland under grant number SFI/12/RC/2289.

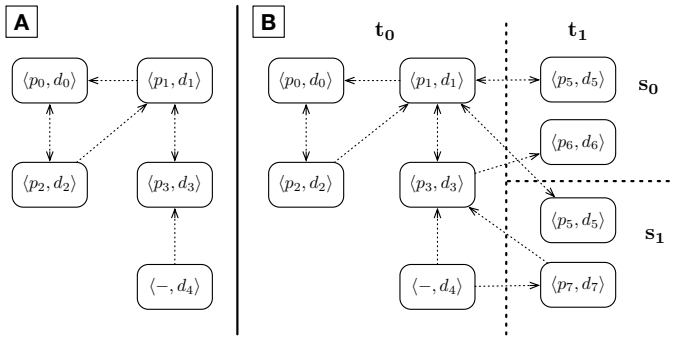

Fig. 1. A: Example graph for an off-line KEP. B: Example graph for a scenario-based anticipatory approach

way exchange by $d_0 \rightarrow p_2, d_2 \rightarrow p_1, d_1 \rightarrow p_0$, a chain starting from an altruistic donor by $d_4 \rightarrow p_3$.

Exchanges involving an arbitrary number of nodes (i.e. k-way exchanges) have been considered in the literature. Dealing with more than three/four pairs is hard in practice, since all the operations often take place simultaneously to avoid withdrawal of donors after their pair received the organ.

A. The Off-line Kidney Exchange Problem

Most works in the literature have focused on finding the best matching for a given graph, i.e. *on the off-line version of the KEP*. In the Operations Research community the problem has been considered in [2], [5], [6], and more recently in [7], [8].

One popular Mathematical Programming model for the KEP is the so-called *cycle formulation*. The model is defined over a directed graph $\langle V, A(V)\rangle$, where V is the set of nodes (pairs and altruistic donors) and $A(V)$ is the set of arcs. We assume that $A(\cdot)$ is a function that maps a set of nodes into a set of arcs, based on their compatibilities. The model requires enumerating the set $\mathcal{C}$ of all cycles that respect the problem. Paths starting from altruistic donors are equated to cycles. Each cycle C_j corresponds to a set of node indices and is associated to a weight $w(C_j)$, equal to its cardinality when the goal is maximizing the number of transplants. The cycle formulation requires to introduce a binary decision variable x_j for each cycle $j \in \mathcal{C}$ such that $x_j = 1$ iff the corresponding exchange has been chosen. The model is then given by:

$$\max z = \sum_{j \in \mathcal{C}} w(C_j) x_j \tag{1}$$

$$\text{s.t.} \sum_{j \in \mathcal{C}, i \in C_j} x_j \leq 1 \qquad \forall i \in V \tag{2}$$

$$x_j \in \{0, 1\} \qquad \forall j \in \mathcal{C} \tag{3}$$

where the objective is to maximize the sum of weights (i.e. the total number of transplantations). Constraint (2) prevents the model from using the same node into more than one cycle.

The model is simple and capable to easily capture complex constraints on the cycle formation (e.g. cardinality restrictions). The main drawback is limited scalability, since the number of cycles is worst-case exponential in the maximum graph cardinality. This issue has been addressed via column generation in [9]–[13]. Recently, alternative compact models have been proposed in [7], [8] to improve scalability without resorting to column generation: the papers contain also pointers to other relevant references for the interested reader.

Algorithm 1: APST1 (adaptation of REGRET)

```
Set score_j = 0 for each cycle C_j in ⟨V_0, A(V_0)⟩
for all scenario s ∈ {1 ... n_s} do
    Solve KEP on graph ⟨V_0 ∪ V_s, A(V_0 ∪ V_s)⟩
    for all cycle C_j in graph ⟨V_0, A(V_0)⟩ do
        if C_j is in the solution then
            score_j = score_j + sol. value
        else
            w_j = w_j − δ
Solve KEP on ⟨V_0, A(V_0)⟩ with the computed scores
```

B. The On-line Kidney Exchange Problem

In the online setting, pairs appear and expire at each time step. Works [7], [10], [14] have taken into account the effect of potential failures, but not of entering pairs. A simulator for the on-line KEP is presented in [15], but the authors still rely on periodic execution of an off-line approach. To the best of our knowledge, the on-line version of the KEP has been targeted using anticipatory algorithms only in [4].

Formally, the on-line KEP is a multi-stage stochastic problem whose exact solution is a *policy tree*, which specifies recursively the best matching for each step and possible uncertain outcome. This approach has clear scalability issues, which are overcome in [4] via *scenario sampling*, based on the ideas from [3] and references therein. A scenario refers here to a set of nodes that may enter in the next h steps, where h is called the look-ahead horizon. A number n_s of scenarios can be obtained by sampling the probability distribution for pair entries (estimated from medical data). All scenarios are considered equally likely after sampling. We refer as V_0 to the nodes currently in the program and as V_s (with $s \in \{1 \dots n_s\}$) to the those entering under scenario s. An example of two simple scenarios with $h = 1$ is shown in Figure 1.

The two best performing algorithms from [4] are based on: 1) sampling a subset of scenarios; 2) using the scenarios to compute scores for the cycles in the current time step; 3) solving a cycle-based off-line KEP for the current time step, with the computed scores. Both algorithms attempt to estimate the expected impact of choosing a cycle in the current time step, but they differ in how the estimate is computed.

The first algorithm in [4] (an adaptation of the REGRETS method from [3]) computes cycle scores *by solving an off-line KEP for each scenario*. Cycles (in the current graph) that are chosen in the solution are rewarded with the value of the solution, cycles that are not chosen receive a fixed penalty δ. We refer to this methods as APST1 (from the initials of the authors): the pseudo-code is reported in Algorithm 1.

The second algorithm in [4] (which shares ideas with the EXPECTATION method from [3]) computes scores *by solving*

Algorithm 2: APST2 (loosely based on EXPECTATION)

Set $score_j = 0$ for each cycle C_j in $\langle V_0, A(V_0)\rangle$
for all cycle C_j in graph $\langle V_0, A(V_0)\rangle$ **do**
  for all scenario $s \in \{1 \dots n_s\}$ **do**
    Solve KEP on $\langle V_0 \cup V_s \setminus C_j, A(V_0 \cup V_s \setminus C_j)\rangle$
    $score_j = w_j +$ sol. value
Solve KEP on $\langle V_0, A(V_0)\rangle$ with the modified weights

an off-line KEP for each cycle and scenario. The cycle scores are obtained by summing the solution values. We refer to this method as APST2 and its pseudo-code is in Algorithm 2.

III. Collective Scenario-Based Algorithm (CSBA)

In the experimentation from [4], both APST1 and APST2 performed considerably better than a non-anticipatory algorithm solving an off-line KEP at each time step. However, both algorithms have a few significant weak points.

First, since there is no closed-form formula for the scores, *they require the use of a cycle formulation for the final KEP*. This prevents the use of the compact models from [7], [10], [14] and *may lead to scalability issues*. This is particularly true for APST2, which also needs to solve a set of optimization problems for each cycle. Second, *none of the two methods correctly estimates (up to the sampling error) the expected number of transplants*. The APST2 scores are proportional to the expected value of choosing a single cycle, but disregard non-linear effects arising when multiple cycles are chosen.

Third, *both APST1 and APST2 sometimes cannot delay exchanges, even if it may have a beneficial effect on the long run*. This is illustrated in Figure 2, which shows solutions for the example from Figure 1 (node descriptions have been replaced with numbers). We assume that the scenarios represent the only possible future outcomes; dark-shaded nodes are included in the matching for the current time step t_0; light-shaded nodes will be included in a matching at t_1 for at least one scenario, while non-shaded nodes have no chance to be included in a matching. A non-anticipatory algorithm would include at t_0 as many nodes as possible, leading to the solution in Figure 2A. APST2 always computes positive scores, and therefore its final KEP will necessarily choose to include the chain $4 \to 3$. This yields the solution from Figure 2B, which is sub-optimal in terms of expected value. In this simple example APST1 would assign a negative score to the chain $4 \to 3$ and to the cycle $1 \leftrightarrow 3$, leading to Figure 2C and the optimal expected value. However, this depends in general on the (heuristic) penalty value δ, and may fail to happen in a more complex example.

We propose to address all these issues by *solving a single KEP on a graph constructed using all scenarios*, but with modified constraints. The final matching is given by all exchanges in the solution that are defined solely over nodes from the current time step. Indeed, we simply propose to employ the classical Sample Average Approximation scheme for two-stage stochastic programs (see e.g. [16]), which to the best of our knowledge has never been applied in this context.

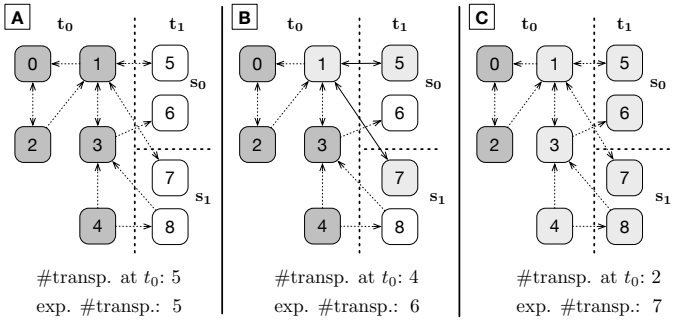

Fig. 2. A: A non-anticipatory solution. B: The solution returned by APST1; C: the solution with the best expected value (returned by APST2 in this case)

The CSBA algorithm is not tied to a specific KEP formulation, and hence we start by providing an abstract description. Let x be the set of problem variables (both for the current time step and the scenarios). Let F be the feasible set for such decisions. Finally, let $u_{s,i}(x)$ be a function denoting the number of times node 1 is used (i.e. participates in an exchange), assuming that scenario s occurs. Then our abstract model is given by:

$$\max z = \frac{1}{n}\sum_{s=1}^{n_s}\sum_{i\in V_s} u_{s,i}(x) \tag{4}$$

$$\text{s.t. } u_{s,i}(x) \le 1 \qquad \forall s = 1 \dots n_s, \forall i \in V_0 \cup V_s \tag{5}$$

$$x \in F \tag{6}$$

which assumes equally likely scenarios. Equation (4) is the expected total amount of transplants, while Equation (5) states that, considering each scenario individually, the same node cannot participate into two exchanges.

In the case of the cycle formulation, the approach can be instantiated by introducing a variable for each cycle (including those from the scenarios). Specifically, let $\mathcal{C}_0$ be the set of cycles in graph $\langle V_0, A(V_0)\rangle$ and let $\mathcal{C}_s$ be the set of cycles in $\langle V_0 \cup V_s, A(V_0 \cup V_s)\rangle$, excluding the cycles already in $\mathcal{C}_0$. Let x_s, j be the variable associated to $C_{s,j}$. Then we have:

$$u_{s,i}(x) = \sum_{\substack{C_{0,j}\in\mathcal{C}_0,\\ i\in C_{0,j}}} x_{0,j} + \sum_{\substack{C_{s,j}\in\mathcal{C}_s,\\ i\in C_{s,j}}} x_{s,j} \tag{7}$$

I.e. a node is used when scenario s occurs if it participates to a cycle in $\mathcal{C}_0$ or to a cycle in $\mathcal{C}_s$. By substituting we get:

$$\max z = \sum_{j\in\mathcal{C}_0} w(C_{0,j})x_{0,j} + \frac{1}{n_s}\sum_{s=1}^{n_s}\sum_{i\in\mathcal{C}_s} w(C_{s,j})x_{s,j} \tag{8}$$

$$\text{s.t.} \sum_{j\in\mathcal{C}_0, i\in C_{0,j}} x_j \le 1 \qquad \forall i \in V_0 \tag{9}$$

$$Eq.\ (7) \qquad \forall s = 1 \dots n_s, \forall i \in V_0 \cup V_s, \exists C_{s,j} \in \mathcal{C}_s : i \in C_{s,j} \tag{10}$$

$$x_{s,j} \in \{0,1\} \qquad \forall s = 0 \dots n_s, \forall j \in \mathcal{C}_s \tag{11}$$

where terms related exclusively to cycles in the current time step have be collected via factorization. This improves the

model performance at the expense of readability. We stress that the approach does not apply exclusively to the cycle formulation, but the cycle formulation provides a good instantiation example due to its simplicity.

IV. Experiments

Since the problem instances from [4] were not available, we resorted to generating new ones following the same general rules. In particular, we consider two experimental setups: in the first one (referred to as "small") we assume that 5 pairs arrive each month for over 31 months. In the second setup (referred to as "big") 10 pairs arrive each month for over 51 months. For each setup, we generate a global pool of pairs with blood types, PRA values, and ages following the distributions from [17]. Overall, we have 158 pairs (11 altruistic donors) and 4,086 edges in the small setup, and 510 pairs (25 altruistic donors) and 15,400 edges in the big setup. Fixed drop-off dates for each pair are generated based on 10-year survival percentages from [18]. Our data is available on-line [1].

Each simulation requires to sample (at each time step) a set of entering pairs from the pool, uniformly at random and without reinsertion. The pairs in each scenario are sampled in the same fashion, except that 1) reinsertion is used and 2) pairs are sampled for each future time step within the look-ahead horizon (or the end of the simulation if it comes earlier).

We implemented the cycle-based model from Section III in Python, using Numberjack [19] as a modeling front-end and CPLEX as a back-end. The same tools we used to implement the APST1 and APST2 algorithms. We also include in our comparison: 1) a baseline approach that solves an off-line KEP for each step; and 2) an oracle that solves a single off-line KEP including all pairs entering the program in a simulation (and disregarding drop-off dates). The "off-line" approach is representative of the current practice, and the oracle provides an optimistic bounds on the performance of any on-line algorithm. All the tests have run on Intel Xeon E5-2640 machines. We consider all transplants equally worthy.

A. Effect of lookahead and number of scenarios

Similarly to [4], we tested different number of scenarios and lookahead horizon values. Due to space limitations, we report only a subset of there results in Tables I and II. Every table column corresponds to a different algorithm and each row to a different configuration. Each row is denoted by $hx_1\_n_sx_2$, where x_1 is the lookahead horizon and x_2 is the number of scenarios. For the small setup $x_1 \in \{2, 4, 7\}$ and $x_2 \in \{5, 10, 15\}$, and for the big setup $x_1 \in \{5, 10, 20\}$ and $x_2 \in \{10, 20, 50\}$. Each cell reports the average number of transplants (saved lives) over 10 runs and the standard deviation. The optimal horizon and number of scenarios for each approach is reported in bold. The oracle and the baseline approaches do not actually make use of scenarios, but are reported on each row for reference.

In Table I we show the results for the small (31-months) setup. CSBA algorithm outperforms the other scenario-based methods, saving on average 1.6 more lives than APST1, 3.4 more than APST2, 8 more than the baseline, and only 5.9 less than the oracle. Note that even a small improvement can be significant when measured in human lives. Unlike in [4], we found that APST1 performed better than APST2: we suspect this may be due to subtle differences in instance generation.

Figure 3 shows the evolution of the cumulative number of saved lives for each algorithm. Interestingly, CSBA method lags behind for the first months, and then rapidly improves: this matches the intuition from Figure 2 that a correct estimate of the expected value may delay some transplants to maximize the long term performance. This behaviour seems to pay off, since the method eventually takes over the competitors.

TABLE I
31-MONTHS SETUP

Lookahead/ Scenarios	CSBA	APST2	APST1	Baseline	Oracle
h2_n_s5	50.3 ± 5.84	48.7 ± 6.24	49.5 ± 5.40	44.7± 5.36	57.1 ± 7.38
h2_n_s10	48.5 ± 4.17	46.5 ± 4.58	49.0 ± 5.09	42.8± 5.79	56.0 ± 5.03
h2_n_s15	52.8 ± 6.16	49.4 ± 5.29	51.6 ± 5.08	44.5± 3.82	60.1 ± 5.10
h4_n_s5	49.3 ± 4.10	45.7 ± 4.60	48.1 ± 4.70	43.3± 5.56	54.7 ± 6.00
h4_n_s10	52.7 ± 5.02	49.1 ± 5.00	50.7 ± 4.79	42.7± 5.83	57.5 ± 3.95
h4_n_s15	55.5 ± 8.62	**52.3 ± 8.24**	**54.7 ± 8.61**	47.4± 8.33	61.1 ± 9.82
h7_n_s5	**56.4 ± 5.44**	51.7 ± 4.56	52.9 ± 5.31	47.2± 2.99	61.9 ± 5.90
h7_n_s10	54.4 ± 8.94	49.6 ± 7.72	51.4 ± 7.56	45.2± 5.19	59.8 ± 9.07
h7_n_s15	53.3 ± 7.82	49.3 ± 6.98	50.6 ± 6.97	43.7± 5.44	57.9 ± 8.01

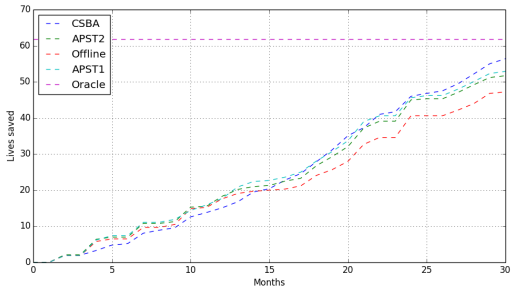

Fig. 3. Trend of the algorithms (31 months setup)

TABLE II
51-MONTHS SETUP

Lookahead/ Sampling	CSBA	APST2	APST1	Baseline	Oracle
h5_n_s10	142.2 ± 10.43	133.8 ± 11.12	140.2 ± 12.89	117.6 ± 9.00	172.5 ± 12.53
h5_n_s20	151.6 ± 10.79	**144.6 ± 9.39**	149.2 ± 11.91	125.5 ± 10.36	186.3 ± 12.87
h5_n_s50	151.1 ± 12.16	142.8 ± 9.48	**150.1 ± 12.61**	123.1 ± 7.67	180.4 ± 12.43
h10_n_s10	149.5 ± 11.79	137.7 ± 8.48	146.3 ± 8.48	122.1 ± 8.53	177.5 ± 11.30
h10_n_s20	152.8 ± 10.54	138.2 ± 8.84	143.2 ± 7.50	119.2 ± 8.55	182.4 ± 11.77
h10_n_s50	158.4 ± 11.56	139.8 ± 7.13	148.0 ± 10.26	123.9 ± 9.62	184.6 ± 11.58
h20_n_s10	154.9 ± 8.63	139.1 ± 7.87	146.9 ± 7.36	124.2 ± 7.08	172.6 ± 8.53
h20_n_s20	**160.6 ± 9.23**	141.7 ± 9.66	149.7 ± 10.62	122.3 ± 5.34	180.6 ± 11.76
h20_n_s50	-	-	-	-	-

In Table II we present the results of the 51-months setup. CSBA algorithm saves on average 12.9 lives more than APST2, 5.9 more than APST2, 30.4 more than the baseline, and 26 less than the oracle. The best configuration for each approach is highlighted in bold. For the configuration $h20\_n_s50$ the solution times for some of the approaches were too large for practical use. Figure 4 shows the evolution of the cumulative number of saved lives for all algorithms, which shows a trend similar to that of Figure 3.

[1] In case the paper is accepted

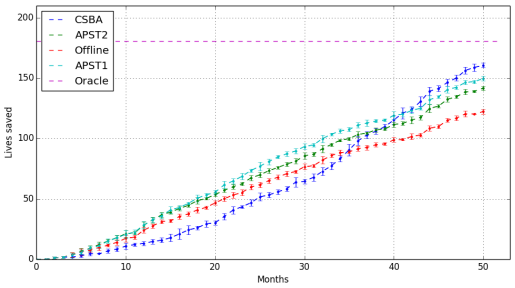

Fig. 4. Trend of the algorithms (51 months setup)

TABLE III
COMPARISON OF SOLUTION TIMES (31-MONTHS SETUP)

Lookahead/Sampling	CSBA	APST2	APST1
h2_n_s5	0.03942	2.17652	0.05244
h2_n_s10	0.04707	1.81701	0.08525
h2_n_s15	0.05912	2.72386	0.13198
h4_n_s5	0.07757	1.63905	0.07198
h4_n_s10	0.15664	4.52681	0.14688
h4_n_s15	0.1469	5.83408	0.22714
h7_n_s5	0.32522	3.81995	0.13239
h7_n_s10	0.81852	4.25008	0.25438
h7_n_s15	0.99315	7.42773	0.3321

B. Solution times

In Table III we report the solution time for all scenario-based algorithms (not including the time for preprocessing, graph construction, and scenario generation). Each value is measured in seconds and represents the average of 10 runs for each configuration. Let us assume that for each step we have m scenarios and n cycles. The ATSP2 algorithm has to solve $m \times n$ sub-problems per time step for adjusting the scores, plus the last problem to compute the solution. ATSP1 solves m sub-problems for the scores, plus one for the final solution. CSBA algorithm needs to solve a single problem per time step. On the other hand, the APST2 and APST1 subproblems are defined using a single scenario, while our algorithm needs to consider all scenarios at the same time.

Since the KEP is NP-hard, solving multiple smaller sub-problems is in theory preferable to solving a single large one. In practice, however, our method is always faster (in the considered benchmark) than APST2, and sometimes even faster then APST1 (which was designed with efficiency as one of its main goals). This is due to the effectiveness of modern ILP solvers, and to the fact that the number of cycles in a graph (and hence of the subproblems that APST2 needs to solve) grows exponentially with the size.

V. CONCLUSIONS

We presented a simple and general method to build an anticipatory algorithm for the on-line KEP from a model for the off-line version of the problem. The method allows us to obtain an expected value estimate that is correct up to the sampling error. The method improves over the state of the art in terms of number of saved lives and has reasonable scalability.

TABLE IV
COMPARISON OF SOLUTION TIMES (51-MONTHS SETUP)

Lookahead/Sampling	CSBA	APST2	APST1
h5_n_s10	0.82089	24.17977	0.82821
h5_n_s20	2.06233	65.99152	2.41063
h5_n_s50	4.21478	127.37255	4.31868
h10_n_s10	5.84768	65.62082	2.56756
h10_n_s20	11.69444	89.10698	4.39736
h10_n_s50	75.05608	170.21087	7.55183
h20_n_s10	27.96764	66.508	5.30497
h20_n_s20	117.4085889	166.04401	11.94762

REFERENCES

[1] J. P. Dickerson and T. Sandholm, "Liver and multi-organ exchange," in *American Journal of Transplantation*, vol. 13, 2013, pp. 272–273.

[2] D. F. Manlove and G. O'Malley, "Paired and altruistic kidney donation in the UK: algorithms and experimentation," *ACM Journal of Experimental Algorithmics*, vol. 19, no. 1, 2014.

[3] P. V. Hentenryck and R. Bent, *Online stochastic combinatorial optimization*. The MIT Press, 2009.

[4] P. Awasthi and T. Sandholm, "Online stochastic optimization in the large: Application to kidney exchange." in *Proc. of IJCAI*, vol. 9, 2009, pp. 405–411.

[5] V. Mak-Hau, "On the kidney exchange problem: cardinality constrained cycle and chain problems on directed graphs: a survey of integer programming approaches," *Journal of combinatorial optimization*, vol. 33, no. 1, pp. 35–59, 2017.

[6] R. Anderson, I. Ashlagi, D. Gamarnik, and A. E. Roth, "Finding long chains in kidney exchange using the traveling salesman problem," *Proceedings of the National Academy of Sciences*, vol. 112, no. 3, pp. 663–668, Jan. 2015.

[7] F. Alvelos, X. Klimentova, A. Rais, and A. Viana, "A compact formulation for maximizing the expected number of transplants in kidney exchange programs," in *Journal of Physics: Conference Series*, vol. 616, no. 1. IOP Publishing, 2015, p. 012011.

[8] J. P. Dickerson, D. F. Manlove, B. Plaut, T. Sandholm, and J. Trimble, "Position-indexed formulations for kidney exchange," in *Proc. of EC*. ACM, 2016, pp. 25–42.

[9] D. J. Abraham, A. Blum, and T. Sandholm, "Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges," in *Proc. of EC*, 2007, pp. 295–304.

[10] J. P. Dickerson, A. D. Procaccia, and T. Sandholm, "Failure-aware kidney exchange," in *Proc. of EC*. ACM, 2013, pp. 323–340.

[11] K. M. Glorie, J. J. van de Klundert, and A. P. Wagelmans, "Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price," *Manufacturing & Service Operations Management*, vol. 16, no. 4, pp. 498–512, 2014.

[12] X. Klimentova, F. Alvelos, and A. Viana, "A new branch-and-price approach for the kidney exchange problem," in *Proc. of ICCSA*. Springer, 2014, pp. 237–252.

[13] B. Plaut, J. P. Dickerson, and T. Sandholm, "Fast optimal clearing of capped-chain barter exchanges." in *Proc. of AAAI*, 2016, pp. 601–607.

[14] J. P. Pedroso, *Maximizing Expectation on Vertex-Disjoint Cycle Packing*. Cham: Springer International Publishing, 2014, pp. 32–46.

[15] N. Santos, P. Tubertini, A. Viana, and J. P. Pedroso, "Kidney exchange simulation and optimization," *Journal of the Operational Research Society*, vol. 68, no. 12, pp. 1521–1532, Dec 2017.

[16] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[17] S. L. Saidman, A. E. Roth, T. Sönmez, M. U. Ünver, and F. L. Delmonico, "Increasing the opportunity of live kidney donation by matching for two-and three-way exchanges," *Transplantation*, vol. 81, no. 5, pp. 773–782, 2006.

[18] J. P. Dickerson, A. D. Procaccia, and T. Sandholm, "Dynamic matching via weighted myopia with application to kidney exchange." in *Proc. of AAAI*, vol. 2012, 2012, pp. 98–100.

[19] N. V. 1.1.0. [Online]. Available: http://numberjack.ucc.ie/